TECHNOSOFT
Instill Excellence

Table of Contents

1. Getting Started with "TS JQuery Form Plugin" 3
2. External Dependencies: 3
3. How to Use "TS JQuery Form Plugin": 3
  3.1 Include CSS file: 3
  3.2 Write an HTML for Plugin: 3
    Step 1 – Place Container for Plugin 3
    Step 2 – Write an HTML Inside Container 4
    Step 3 – Place Save/Cancel Button 6
  3.3 Include JavaScript file: 7
  3.4 JavaScript to Invoke Plugin: 7
4. Plugin Options: 8
  5.1 Theme for Plugin 8
  4.2 Form Layout 9
  4.3 Change Header Style 10
  4.4 Quick Navigation among Pages/Slides 10
    4.4.1 Slides/Pages Linking 11
  4.5 Use Icons Instead of Save and Cancel Buttons 11
  4.6 Show Cancel Button 12
  4.7 Show Save Button 12
  4.8 Form Data Submission 12
    4.8.1 Form Data Submission (ASP.NET MVC) 13
  4.9 Set Plugin Height 14
  4.10 Set Plugin Width 15
  4.11 Group Controls 15
  4.12 Table/Grid Controls 16
  4.13 Hide Navigation Symbols Tray 19
  4.14 Plugin Public Functions 19
    4.14.1 $.fn.SetJformWidth 19
    4.14.2 $.fn.SetJformColumnWidth 19
    4.14.3 JformLabelAlignment 20
    4.14.4 tableToJSON 20
  4.15 Set Field Highlight Style 20
    4.15.1 Field Highlighter Options 21

1. Getting Started with "TS JQuery Form Plugin"

TS JQuery Form Plugin allows user to develop a decent looking web-form where the plugin will cater the alignment and basic design. There are various options such as control and their respective label positioning, control grouping and also have the ability change the form interface to wizard.

2. External Dependencies:

Following are the files which need to be added to use the plugin:

- jquery-1.8.3.js : The file is added to the package. Latest version can also be used.
  Visit **http://www.jquery.com** to get the latest version

- json2.js : The file is added to the package. Latest version can also be used.
  Visit **https://github.com/douglascrockford/JSON-js/blob/master/json2.js** to get the latest version

3. How to Use "TS JQuery Form Plugin":

Follow the procedure described below to use the plugin functionality

3.1 Include CSS file:

Include the Plugin's CSS file as shown in code example below:

Code sample:

```
<link rel="stylesheet" href="css/jquery.ts-jform.0.3.css" type="text/css" />
```

There is another CSS file "**jquery.ts-jform-ie-fix.css**" in a package, there is no need to include this file as Plugin will detect the IE Browser version, it will be loaded by dynamically against IE 6, 7 or 8

3.2 Write an HTML for Plugin:

Plugin takes an html as an input, which needs to be according to the format which is described below:

Step 1 – Place Container for Plugin

Place the container inside <form> tag; Plugin will use the container div and place the generated form html inside it

Code sample:

```
<div id="jFormContainer" data-heading="Registration"></div>
```

- ***data-heading:*** this data attribute use for form heading

Step 2 – Write an HTML Inside Container

For section or pages fieldset should be placed inside container div. Code example given below is for single page and single column structure

Code sample:

```
<div id="jFormContainer" data-heading="Registration">
     <fieldset data-heading="Login Information">
          <div>
            <input data-label="User Name" type="text" />
            <input data-label="Password" type="password" />
            <input data-label="Confirm Password" type="password" />
          </div>
     </fieldset>
</div>
```

JForm (one column form)

NOTE:

- <fieldset> inside container represents the page/section of a form
- inside <fieldset> represent the a column
- Plugin will apply styling on control as well such as width, height etc,
- Incase inline styling or CSS class is applied on controls then plugin will not apply its styling and adjust the form accordingly

Similarly, for more than one page or section another field set can be added, code example below will generate two page or section and both having single column structure

Code sample:

```
<div id="jFormContainer" data-heading="Registration">
    <fieldset data-heading="Login Information">
          <div>
            <input data-label="User Name" type="text" />
            <input data-label="Password" type="password" />
            <input data-label="Confirm Password" type="password" />
          </div>
    </fieldset>
    <fieldset data-heading="Personal Details">
        <div>
            <input data-label="First Name" type="text" />
            <input data-label="Last Name" type="text" />
        </div>
    </fieldset>
</div>
```

Furthermore, if multiple column structure is required, add div inside fieldset and plugin will generate new column inside page or section and adjust the form accordingly, see the code and figure below

Code sample:

```
<div id="jFormContainer" data-heading="Registration">
     <fieldset data-heading="Login Information">
          <div>
            <input data-label="User Name" type="text" />
            <input data-label="Password" type="password" />
            <input data-label="Confirm Password" type="password" />
          </div>
    </fieldset>
    <fieldset data-heading="Personal Details">
          <div>
            <input data-label="First Name" type="text" />
            <input data-label="Last Name" type="text" />
          </div>
          <div>
            <input data-label="Phone" type="text" />
            <select data-label="country">
                 <option value="USA">USA</option>
            </select>
          </div>
    </fieldset>
</div>
```

JForm (two columns form)

- ***data-heading:*** Use data-heading attribute for setting page heading which will be shown in the header area of the page, see the code below

Code sample:

```
<fieldset data-heading="Login Information">
```

- ***data-label:*** data-label attribute is very important attribute, as this is the attribute using which label can be assigned to the controls. Add data-label attribute to control tag, plugin will dynamically use it as label, all the formatting and alignment will be handled by the Plugin itself

Code sample:

```
<input data-label="Phone" type="text" />
```

Step 3 – Place Save/Cancel Button

Plugin allows users to place save and cancel button and gives leverage to submit data, add save and cancel button inside plugin container div as shown in code below:

Note: Add data attribute **data-button-type** to distinguish between "**submit"** and "**cancel"** buttons, and must define submit button type as shown below:

Code sample:

```
<div id="jFormContainer" data-heading="Registration">
   <!--Place your save/cancel button HTML here -->
   <input id="btnSave" data-button-type="submit" type="submit" value="Save" />
   <input id="btnCancel" data-button-type="cancel" type="button" value="Cancel" />
</div>
```

Note:

- Plugin will handle the save and cancel button layout.
- if no functionality is applied, plugin will apply default functionality to “Cancel” button i.e. it will clear all the form fields,
- By default cancel button is hidden, for further details please see “useCancelButton” Plugin option
- Plugin submit data using JSON format, see “submitUrl” option
- For successful data submission add ***id*** or ***name*** attribute to control tag, see the code below

Code sample:

```
<input data-label="First Name" id="txtFirstName" name="txtFirstName" type="text" />
```

3.3 Include JavaScript file:

Include the Plugin’s JS file in head tag of the form as shown in code example below:

Code sample:

```
<script type="text/javascript" src="js/jquery.ts-jform.min.js"></script>
```

3.4 JavaScript to Invoke Plugin:

Now, we are ready to use “TS JQuery Form Plugin”, Place the code sample below in form to invoke the plugin

Code sample:

```
<script type="text/javascript" language="javascript">
        $(document).ready(function(){
            $("#jFormContainer").tsJform();
        });
</script>
```

Note:

- In code example above JQuery $(document).ready function is used, JQuery anonymous function or any other method as per requirement can be used to invoke the plugin

4. Plugin Options:

5.1 Theme for Plugin

Built-in themes can be applied on plugin generated form. ***themeLayout*** is the property for applying theme on plugin generated form.
There are three built-in options i.e. blue, green and brown.

Default: blue

Code sample:

```
$("#jFormContainer").tsJForm({
   themeLayout: "green" // blue, green or brown
});
```

Plugin Theme(green)

If it is required to create custom theme, go to file ***css/jquery.ts-jform.0.3.css*** and see group of theme classes for green, blue and brown etc. create own set of classes with specifying name in the end of their

name. Specify the newly created theme name against ***themeLayout*** property; plugin will load the newly custom theme

4.2 Form Layout

Plugin gives the option to change the web form layout to wizard or full page. ***jFormType*** is the property for that.

Default: wizard

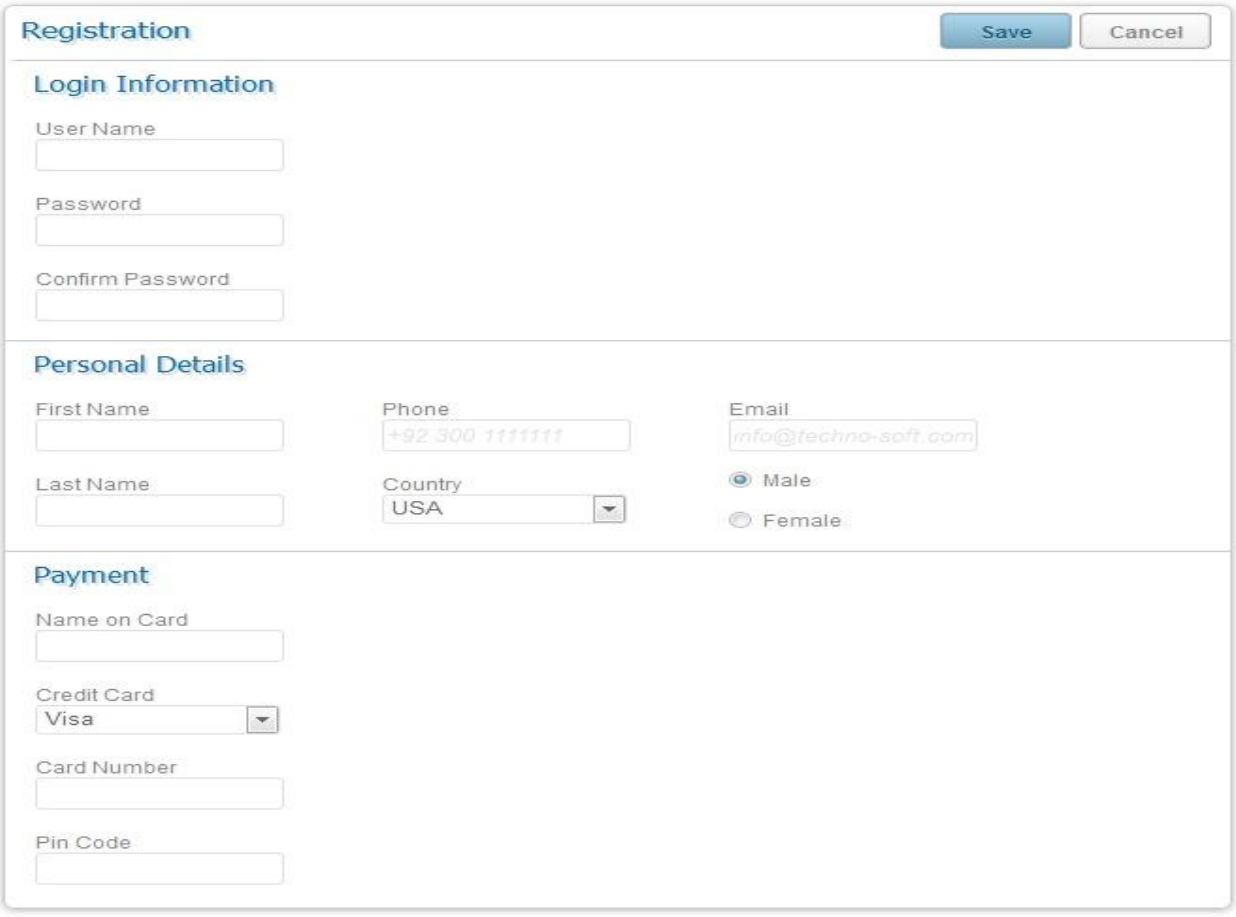
- ***fullpage***:
  Will convert the whole HTML to just one page, no forward/backward/jump to slide options are available

- ***wizard***:
  Will convert the HTML to slides, forward/backward/jump to slide options are available

Code sample:

```
$("#jFormContainer").tsJForm({
   jFormType: "fullpage" // fullpage or wizard
});
```

Web page layout (fullpage):



Page layout (wizard):

4.3 Change Header Style

By using "**useHeaderLayout**" property, plugin can change the header background (i.e. make a main application header) according to the selected theme. There are two options i.e. yes or no

Default: no

Code sample:

```
$("#jFormContainer").tsJForm({
    useHeaderLayout: "yes" // yes or no
});
```

Plugin header (blue header layout):

4.4 Quick Navigation among Pages/Slides

In Wizard layout, Plugin gives the option to navigate directly to any specific for that plugin creates navigation symbols against each page/section

"**navigationSymbol**" is the property for the symbols. Options for this property are mentioned below:

Default: circle

circle: this will convert navigation symbol to circles
square: this will convert navigation symbol to squares
none: this will hide the navigation symbols

Code sample:

```
$("#jFormContainer").tsJForm({
    navigationSymbol: "square" // square, circle or none
});
```

Navigation Symbols(square):

4.4.1 Slides/Pages Linking

Plugin provides a facility to navigate with in document if the page is height large enough and jform is in **"fullpage"** mode. It could be helpful in finding the desired location of the form if form has too much content in it. Clicking on navigation symbol simply scroll to the desired location.

4.5 Use Icons Instead of Save and Cancel Buttons

Based on requirement icons can also be placed against save and cancel button, all it need is url/path of an icon to replace the button. "**cancelIconUrl**" and "**saveIconUrl**" are the property for that. Code sample is given below

Default: empty (no icons)

Code sample:

```
$("#jFormContainer").tsJForm({
    cancelIconUrl: "css/images/cancel.png",
    saveIconUrl: "css/images/save.png"
});
```

4.6 Show Cancel Button

By default plugin does not show cancel button. If the cancel button is required on form place it html as explained in start of the manual and set "**useCancelButton**" property to true.

Default: false (hide the button)

Code sample:

```
$("#jFormContainer").tsJForm({
   useCancelButton: true // true or false
});
```

4.7 Show Save Button

By default plugin show the save button if the html is properly placed as explained in start of the manual. If the save button is not required on form, set "**useSaveButton**" property to false.

Default: true

Code sample:

```
$("#jFormContainer").tsJForm({
   useSaveButton: true // true or false
});
```

4.8 Form Data Submission

Plugin supports submission of data as well, it requires target URL on which it can submit the data. "**submitURL**" is the property exposed for mentioning the target URL. Another option is introduced to use plugin's submission or user's own submission events behind submit/save/clear/cancel buttons. Use this option "**allowJsonSubmission**" to make it working. By default its **"true"** and will be used plugin's submission, setting to "**false"** will fire the submission events of user's code implementation.

Code example below is written in php. **"submitUrl"** property is set to "**registration.php**" and in target form data is being saved in "**output.txt**".

Note: plugin submits the data in JSON format

Code sample:

```
$("#jFormContainer").tsJForm({
   submitUrl: "registration.php", // Target source page url
   allowJsonSubmission: false
});
```

Sample Code (PHP - registration.php):
Example below gets the json and save it in a file name json.txt, and also return json decoded data to AJAX request

Code sample:

```
<?php

  if(isset($_POST["json"]))
   {
      $jsondata = $_POST['json'];
      $data_decode = json_decode(json_encode($jsondata));

      /* for getting table data use this*/
      $jsontabledata = $_POST['jsonTableData'];
      $data_decode_table = json_decode(json_encode($jsontabledata));

      /* writing data to file */
      $file = fopen("json.txt", "w");
      fwrite($file, $jsondata.$data_decode_table);
      fclose($file);

       echo $data_decode;
    }
?>
```

4.8.1 Form Data Submission (ASP.NET MVC)

Plugin can be fully integrated with ASP.NET MVC forms, also supports submission of data to MVC controllers and response back to request. There are some steps to make it working:

1. Create a **Model** class:

Code sample:

```
// a model class for getting json data of form
public class JFormDataModel
{
   public string Name { get; set; }
   public string Value { get; set; }
}
```

2. Passing a "**submitUrl"** like this:

Code sample:

```
$("#jFormContainer").tsJForm({
   submitUrl: "/home/PostJson" // call PostJson function in Controller
});
```

3. Writing **"PostJson"**:

Code sample:

```
// a json data posted controller example
[HttpPost]
public ActionResult PostJson(List<JFormDataModel> jsonDataCollection)
{
   string writeJson = string.Empty;

   var dictJson = new Dictionary<string, string>();
   foreach (var jsonObj in jsonDataCollection)
   {
      //Name: controlName, Value: controlValue
      dictJson.Add(jsonObj.Name, jsonObj.Value);
   }

   //now converting all dictionary items to JSON format and save in one string
   writeJson = new JavaScriptSerializer().Serialize(dictJson);

   //writing a json data to text file
   string fileName = "postJson.txt";
   FileInfo fileInfo = new FileInfo(fileName);
   using (StreamWriter sWriter = fileInfo.CreateText())
   {
     sWriter.WriteLine(writeJson);
   }

  //response back success to .Ajax request
  return Json({ success = "success" }, JsonRequestBehavior.AllowGet);
}
```

Code example above is written in ASP.NET MVC. **"submitUrl"** property is set to **"/home/PostJson"** and in target form data is being saved in "**postJson.txt**".

4.9 Set Plugin Height

Plugin gives the option to set its height as well as it can also dynamically set the height based on the content. "**jFormHeight**" is the property for this purpose. There are multiple options which are mentioned below:

Default: auto

Options:

- ***auto:*** plugin will calculate height and fix all page/slide height
- ***autofit:*** Height of all pages/slide will be calculated dynamically, based on how many controls each column contains
- ***Custom height value:*** This will fix the height of the plugin. Note that numeric value will be applied in pixels

Code sample:
```
$("#jFormContainer").tsJForm({
   jFormHeight: "autofit" // auto, autofit or numeric value
});
```

OR,

Code sample:
```
$("#jFormContainer").tsJForm({
   jFormHeight: 400 // auto, autofit or numeric value
});
```

4.10 Set Plugin Width

Plugin gives the option to set its width as well as it can also dynamically set the width based on the content or number of columns. "**jFormWidth**" is the property for this purpose. There are multiple options which are mentioned below:

Default: auto

Options:

- ***auto:*** plugin will calculate the width of plugin dynamically.
- ***Custom width value:*** This will fix the width of plugin, value will be applied in pixels

Code sample:
```
$("#jFormContainer").tsJForm({
     jFormWidth: 800 // auto or numeric value
});
```

4.10 Group Controls

Plugin gives the support to place controls in group container and highlight the whole group. Insert controls in a ** ** and the heading of group control will be access from its **data-label-group** attribute The HTML should be like this:

Code sample:
```
<div data-label-group="Gender">
    <input type="radio" name="Gender" checked="checked" data-label="Male" />
    <input type="radio" name="Gender" data-label="Female" />
</div>
```

Group controls will be rendered as shown below see highlighted area:

4.11 Table/Grid Controls

Plugin gives the support to place table/grid controls in form and highlight the whole table. Insert **<table>** tag in a ** ** and the heading of table control will be taken from its **data-label** attribute also assign **"ts-jform-table"** class to **<table>.** The table will be automatically rendered at end of the slide. The HTML should be like this:

Code sample:

```
<div>
    <table class="ts-jform-table" data-label="TableHeading">
      <thead>
         <tr>
            <th></th>
            <th></th>
         </tr>
      </thead>
      <tbody>
         <tr>
           <td></td>>
           <td></td>
         </tr>
      </tbody>
   </table>
</div>
```

Note: The **<table>** tags can't be fitted in column so wherever you place **<table>** in section, the plugin will always render the **<table>** at end of that slide. Plugin also provides add/edit/delete functionality for table control. It just need to call a function and pass its table id, call it after invoking plugin.

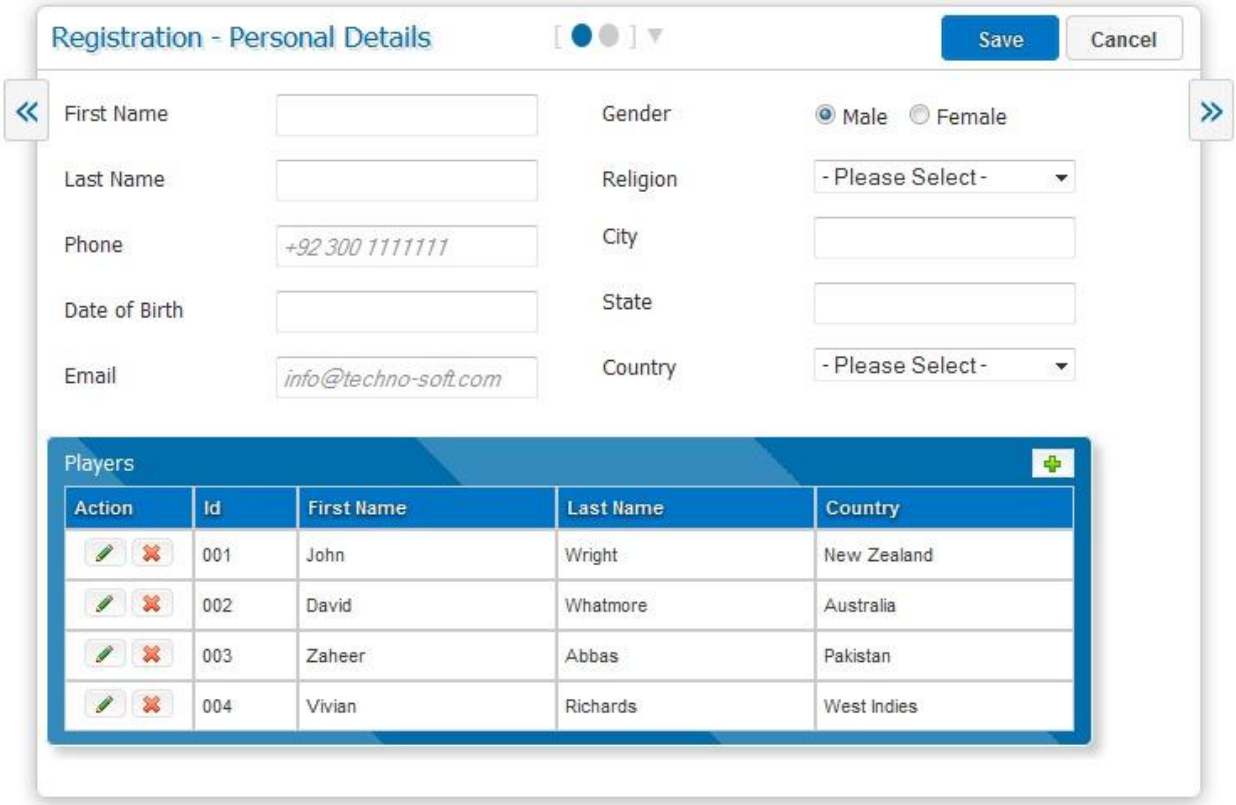
Code sample:

```
$(document).ready(function () {

    $("#jFormContainer").tsJForm();

    // for table add/edit/delete functionality, call it like this
    SetupTableControl("tblPlayers");
});
```

A sample table rendering in form would be this:

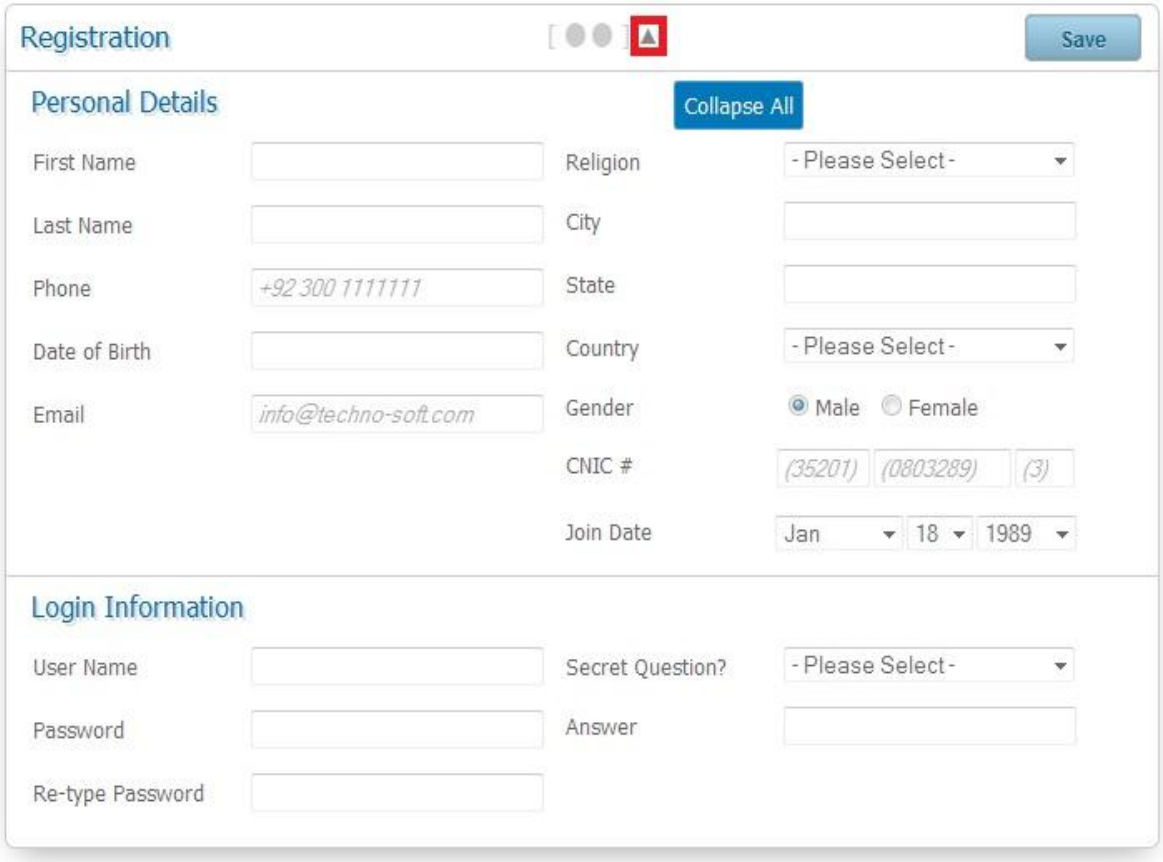


4.12 Expand/Collapse jForm

Plugin has an option to toggle between **"fullpage"** and **"wizard"** jform type. It provides an arrow with the navigation symbols which expands/collapses the jform respectively

For expanding/collapsing jform use this button/arrow:

After expanding, jform would be look alike:



4.13 Hide Navigation Symbols Tray

Plugin provides an option to hide/show the navigation symbol tray, use this property "**hideNavigationTray"** to get the functionality working.

Default: true

4.14 Plugin Public Functions

Plugin provides some public functions that user can call them to align jform perfectly. Each one is listed below with **"how to"**:

4.14.1 $.fn.SetJformWidth

This function accepts custom width value in integer. It will be used to set overall width of jform and the other containers that are inside to make it work smoothly

Code sample:

```
$.fn.SetJformWidth(755); // call it this way or either

$("#jFormContainerDiv").tsJform({
    jFormHeight: "autofit",
    jFormType: "fullpage"
}).SetJformWidth(755); //or like that
```

4.14.2 $.fn.SetJformColumnWidth

User can set every column width in each slide to align jform perfectly. This function accepts three arguments:

1. slideNumber
2. slideColumnNumber
3. columnWidth

For example, a slide number two and its column number one need to be aligned with the rest of jform then how to call this function shown below:

Code sample:

```
$.fn.SetJformColumnWidth(2, 1, 380);
```

4.14.3 JformLabelAlignment

User can set the width and other CSS properties of label of control to align with the other controls in jform. This function primarily accepts CSS properties object as an argument and automatically apply on control's label.

For example, a control id **"txtOne"** label is not aligned with the rest of form controls, and then calls this function like:

Code sample:

```
$("#txtOne").JformLabelAlignment({ "padding-top": "2px", "width": "95px" });
```

4.14.4 tableToJSON

Plugin provides a public function which converts the whole table data to array form and later on can convert to JSON form for submitting to server

For example, a table with id **"tblEmployee"** needs to be converted to JSON format, and then use this function like:

Code sample:

```
var tblArrayForm = $("#tblEmployee").tableToJSON();
var tblJSONFormat = JSON.stringify(tblArrayForm);
```

4.15 Set Field Highlight Style

Plugin has an option to highlight fields. Colors are automatically applied based on the plugin theme. Clicking on field label or control plugin highlights the field and brings the keyboard focus on field control. Figure below is showing the highlighted field

Plugin which controls the field rendering labeling and its layout theming it also perform some other functionality i.e. apply effect on field when hover or click functions are called.

Field Highlighter (By Clicking on a field (input, textarea, select etc) will highlight it**):**

Plugin also give some options to the user to customize field highlight style which are explained further in "**4.15.1 Field Highlighter Options**" section below:

4.15.1 Field Highlighter Options

It uses the color scheme of the current theme set for the plugin, "blue" is the default theme of plugin. Properties are divided into two parts one is related to the field i.e. highlight style on field focus, field border color etc. and the other one is related to label i.e. label positioning and its text alignment. "**fieldHighlighterOptions**" is the main property containing two sub property which are explained below:

fieldHighlighterOptions -> Field Property:

- **Style:**
  Using this property highlight style can be set. Plugin supports three styles which are explained below:

  Default: strip
  Options:
  - ***shadow:*** label position will be at top of control
  - ***gradient:*** label position will be at left side of control
  - ***strip:*** label will be written inside field/control like placeholder

- **Border Color:**
  Plugin gives the leverage to the user to set border color of the highlighted portion
  Default: white
  Options: *any standard color name can be used*

Field property is further explained below using the code samples and figures below:

Code sample:

```
$("#jFormContainer").tsJForm({
    fieldHighlighterOptions: {
        field: { borderColor: "white", style: "strip"}
        // borderColor: Standard color name or code can be used
        // style: shadow, gradient and strip
    }
 });
```

Field Highlighter (strip):

Code sample:

```
$("#jFormContainer").tsJForm({
    fieldHighlighterOptions: {
        field: { borderColor: "white", style: "gradient"}
        // borderColor: any legal color name or value can be used
        // style: shadow, gradient and strip
    }
 });
```

Field Highlighter (gradient):

Code sample:

```
$("#jFormContainer").tsJForm({
    fieldHighlighterOptions: {
        field: { borderColor: "white", style: "shadow"}
        // borderColor: any legal color name or value can be used
        // style: shadow, gradient and strip
    }
 });
```

Field Highlighter (shadow) and theme (brown):

Code sample:

```
$("#jFormContainer").tsJForm({
    themeLayout: "green",
    fieldHighlighterOptions: {
            field: { borderColor: "white", style: "strip"}
        // style color will be applied according to the themelayout
        // borderColor: any legal color name or value can be used
        // style: shadow, gradient and strip
    }
 });
```

Field Highlighter (strip) and theme (green):

fieldHighlighterOptions -> Label Property:

- **position:**
  Using this property label position can be set. Plugin supports three options which are explained below:

  Default: top
  - ***top:*** label position will be at top of control
  - ***left:*** label position will be at left side of control
  - ***inline:*** label will be written inside field/control like placeholder

- **align:**
  Plugin gives the leverage to the user to set label text alignment
  Default: left
  - ***left:*** label will be aligned to left
  - ***right:*** label will be aligned to right
  - ***center:*** label will be aligned to center

Label property is further explained below using the code samples and figures below:

Code sample:

```
$("#jFormContainer").tsJForm({
    fieldHighlighterOptions: {
        label: { align:"left", position:"left"  }
        // align: left or right
        // position: top, left or inline
    }
 });
```

Label positioning (left):

Code sample:

```
$("#jFormContainer").tsJForm({
    fieldHighlighterOptions: {
        label: { align:"left", position:"inline"}
        // align: left or right
        // position: top, left or inline
    }
 });
```

Label positioning (inline):

Code sample:

```
$("#jFormContainer").tsJForm({
    fieldHighlighterOptions: {
        label: { align:"right", position:"left"}
        // align: left or right
        // position: top, left or inline
    }
 });
```

Label alignment (right):

Code sample:

```
$("#jFormContainer").tsJForm({
    fieldHighlighterOptions: {
        label: { align:"right", position:"top"}
        // align: left or right
        // position: top, left or inline
    }
 });
```

Label alignment (right) and position (top):